1-6

SINC-LINK

TIMEX-SINCLAIR USERS CLUB

NEWSLETTER

Toronto, Ontario

PRINTED IN CANADA

B6146

MESSAGE FROM THE PRESIDENT

Dear Members,

In the coming months the executive committee hopes to cut through the slang which is popular these days and give all members a taste of practical uses for their machines. See you at all the meetings.

EXECUTIVE OFFICERS

PRESIDENT: Greg Lloyd
SECRETARY: George Chambers
LIBRARIAN: Martin Mauk
TREASURER: John Roach
NEWS EDITOR: Stan Piotrowski
ACTIVITY DIRECTORS: Brian Hammond, Ian Roberts
MEETING CHAIRMAN: Harold Goodwin
LIASON OFFICER: (out-of-town members): Chris Hart

HARDWARE HINTS

(by Virgil Roman)

OVERHEATING

Many of the ZX-81 owners experienced at least one computer malfunction caused by heat because the chips become extremely hot to allow proper function. To eliminate heat related break-downs, some of the deficiencies, which I have found to exist, must be corrected, such as:

1. poor ventiliation
2. very small heat sink
3. poor thermal conductivity between voltage (heat) regulator and heat sink.

VENTILATION

The lack of ventilation may be resolved by removing the circuit board from the casing; cut or drill ventilating holes (1/4 inch) on the left & right side of the lower half of the computer box just below the key-membrane. Underneath, on the same bottom half, drill or cut 2 groups of slots just beside the original ones. (See fig. 1)

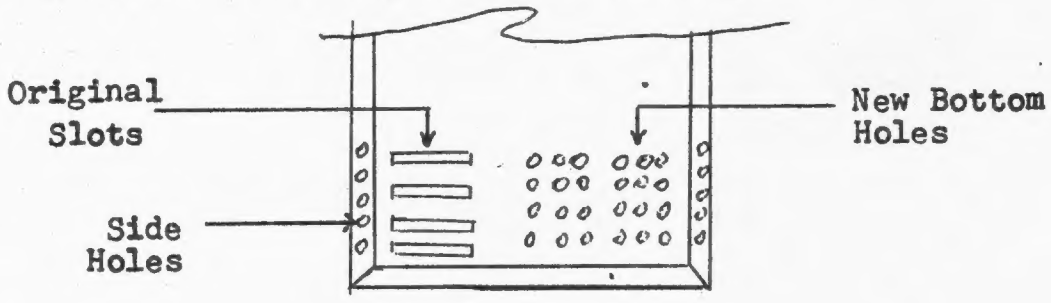

fig. 1

Having done this, purchase the rubber cushion feet from Radio Shack (cat. #64.2342) & apply them in place of the original ones. These feet will raise the computer about 1/2 inch higher from the working surface

leaving enough space for air circulation under the computer, through the ventilating holes and over the heat sink; thus cooling it faster.

HEAT SINK

Before you re-assemble the computer, there is yet something else that you should be done. The original aluminum heat sink the computer is provided with, is too small to cool off the components (chips). Such as I have done, I strongly suggest replacing it with a larger one. My replacement is 14 cm. long by 4 cm. wide occupying the whole space under the key-membrane. (see fig. 2)

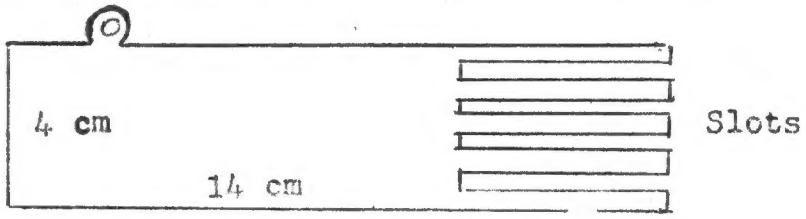

fig. 2

To increase the capacity of the aluminum sink to dissipate the heat, cut 5 or 6 slots on the right side of the aluminum plate and paint with a flat black or sand blast it, BUT DO NOT POLISH IT. All the heat sinks in the larger micro-computers are installed black, not painted black but rather made black through the chemical process known as "anodyzing". This fancy process is out of reach to most of us; thus, we must be contented with the flat paint only. It is known that the colour black has the ability to release heat 20 times greater than that of polished aluminum. It is for this reason that I suggest the flat black paint.

THERMAL CONDUCTIVITY

The point of juction between the regulator and the heat sink is not one of the best. For a fast conductivity of heat, eclectrical, etc., it is necessary that the conductor be uninterrupted. To eliminate the air bubble which causes the interuption at the critical joint (heat sink and regulator), is very simple: just purchase Heat Sink Compound (silicon + Zincoxide) from Radio Shack and before installing the new improved heat sink, smear the surface of contact point.

Having done all the above mentioned changes to my own unit, I am enjoying "cool" computing. Do the same to your computer and it will always run "cool" even when.....your temper runs "high".

Do not be afraid to open the box and experiment with it, there is nothing to damage. (make sure power is always off when doing anything inside. ed.)

GOOD LUCK!!!

ALIGNING DECIMAL POINTS

(by G.F. Chambers)

Often a program will require a column of numbers to be printed. The ZX-81 normally will not align decimal points and this leaves a ragged

appearance. The following subroutine will place a column of figures with their decimal points aligned:

```
9000  PRINT "(7 or so spaces)"(1 TO M-(V > 1) * INT (LN
V/LN 10) + (V <.1)); V
9010  RETURN
```

V = number to be printed
M = sets field width i.e. number of printing positions before the decimal point
M is set earlier in the program. e.g. LET M = 7

To demonstrate the subroutine set-up, this is the following program:

```
10  LET M=7
20  INPUT V
30  GOSUB 9000
40  GOTO 20
9000  (as shown earlier)
9010  RETURN
```

Essentially, line 9000 determines the difference between the number of characters to the left of the decimal in V; subtracts it from M; and adds spaces (sliced from the blank string in line 9000) to V before printing V.

The numbers of spaces in line 9000 should not be less than the value of M.

DECIMAL CONTROL

THE ZX-81 has rather primitive number handling facilities when it comes to printing. If you ask it to print a computed value, you will have no control over the number of figures which will be printed after the decimal point. This can be troublesome especially if it is a money figure you wish to display.

One method of handling this is by means of the following subroutine:

```
9000  LET A = 10 ** N
9010  PRINT INT (V * A)/A
```

Where:
V = the number to be printed
N = the number of digits to follow the decimal point.

Enter the following demonstration program:

```
10  LET N=2
20  INPUT A$
30  LET V=VAL A$
40  GOSUB 9000
50  GOTO 10
9000  as shown
9010  as shown
9000  RETURN
```

Run and enter the decimal values per line 20. Note that on line 9000

the "**" is found on Key 'H' (meaning: to the power of..). Line 20 is used to convert the variable to be treated to the variable 'V' used in the subroutine.

As it stands, line 9010 chops off all unwanted numbers; i.e. it rounds DOWN. If you want the program to round to the nearest number then make line 9010 as follows:

```
9010  PRINT INT (V * A + .5)/ A
```

BASIC PROGRAMMING

it's time again to return to a D-Base type program of 'Name & Address' file. We have seen sofar how to set up and DIMension the file; set up the Menu; add data to the file; list the entire file; delete an entry. Now we want to be able to use the computer to its capabilities which is SEARCHING.

We know that if someone looks at the number 10, you can search through a group of numbers until you find another 10 to match it...very simple. But try that with a long sentence; in our case we have DIMensioned that sentence (string or '$' in computer language) to 88 characters. You can see that it would be a long and tedious job of comparing each character for a match which the computer doesn't mind doing over and over again with great speed.

Therefore it is a simple matter to give the user capabilities to enter data in a string then the computer compares that string to the data file. Enter the following lines which will be explained.

```
5000  FAST
5010  CLS
5020  PRINT AT 8,0;"ENTER NAME TO SEARCH"
5025  SLOW
5030  INPUT S$
5031  S$=S$+"  26 spaces "
5032  S$=LEN S$(1 TO 26)
5035  FAST
5040  LET I=1
5050  IF S$=R$ (I,1 TO 26) THEN GOTO 5100
5060  LET I=I+1
5070  IF I > N THEN GOTO 5090
5080  GOTO 5050
5090  CLS
5095  PRINT AT 12,0;S$
5097  PRINT AT 8,6;"NO MATCHES FOR"; AT 20,6;"PRESS A KEY FOR
MENU"
5096  PAUSE 4E4
5099  GOTO 100
5100  CLS
5105  PRINT AT 3,5;"RECORD NUMBER ";I
5110  GOSUB 20
5120  PRINT AT 20,0;"C=CONTINUE, M=MENU, P=PRINTOUT"
5130  PAUSE 4E4
5140  IF INKEY$="P" THEN GOTO 5200
5150  IF INKEY$="M" THEN GOTO 100
5160  IF INKEY$="C" THEN GOTO 5300
5170  GOTO 5130
5200  COPY
```

```
5210  GOTO 5120
5300  CLS
5320  LET I = I + 1
5330  GOTO 5050
```

Add the following lines to SAVE your program and data:

```
6000  FAST
6010  CLS
6020  PRINT AT 8,0;"ENSURE TAPE RECORDER IS ON"
6030  PRINT,,"THE RECORD MODE THEN PRESS A KEY"
6040  PAUSE 4E4
6050  SAVE "NAME FILE"
6060  GOTO 100
```

Also add these lines:

```
20  PRINT R$(I,1 TO 26)
22  PRINT R$(I,27 TO 49)
24  PRINT R$(I,50 TO 57)
26  PRINT R$(I,58 TO 69)
28  PRINT R$(I,70 TO 76)
30  PRINT R$(I,77 TO 88)
32  PRINT AT 20,8;"PRESS A KEY"
34  PAUSE 4E4
36  RETURN
```

Line 5030 is the data input you wish searched and as it stands, you must enter the search data exactly the same as it was entered originally. Since we have DIMensioned the name as 26 characters, we must pad out the remainder of S$ with spaces. The next line chops off all extra spaces after the name is entered plus the padded out spaces to a length of 26 spaces. For example, if we were searching: "JOHN SMITH", we see that there are 10 characters (including the space between JOHN and SMITH). R$ has set aside the first 26 characters for the name so we add 26 spaces to the data input of S$ which is JOHN SMITH. We now have 10 characters + the 26 added spaces which equals 36 characters...more than we need. The next line then chops off the unnecessary length by typing: S$=LEN S$(1 TO 26), which is the original name of 10 characters + the number of spaces to equal 26 (or 16 spaces).

Note in line 5040 we use: LET I=1 as opposed to a FOR/NEXT loop. The reason is that further on, we have a choice of returning to the menu or continuing our search. If we return to the menu from a FOR/NEXT loop, the computer is still hanging on waiting for the loop to complete itself. This is 'STACKING' in machine code. You may have read somewhere that the computer reserves so much memory for itself forthe STACK. The stack is used by other commands as well such as the GOSUB command. It stores the line it must return to on the stack. The stack has only so much memory so if we break out of the FOR/NEXT loops too often, we run out of 'STACK' memory and the program crashes. Therefore, by simply adding numbers to 'I' we can break out anytime with no worries.

Line 5070 is a check that the searching does not continue beyond the amount of records on file held by the variable 'N' so we don't have unnecessary searching.

Line 5050 checks for a match between the name entered and the data on file. Since this is a very simple D-Base program, you can only search for a name but seeing how the above is done, you should be able to add

your own lines to search other areas such as an address, etc.

The remaining part of the program has been dealt with in the previous Newsletters but you should be able to follow through easily enough.

In the next and following Newsletters, we will go into sorting the files and also expanding the program such as partial searching (i.e. entering partial data instead of the exact data to be searched).

So until the next time, learn and enjoy programming.

MERGING BASIC PROGRAMS

(by G.F. Chambers)

A method to allow a second program to be inserted into the ZX-81 without losing the first program.

Assuming you already have a program in your ZX-81 and wish to interrupt it to write a second small program, enter the follwoing which will (APPARENTLY) NEW the existing program.

POKE 16509,100

The screen will be blank and if you LIST you will darw a blank. You can then enter a new program and RUN it. To retrieve the first program: POKE 16509,0. Before you do this, however, delete the first program line by line; do not delete it by using NEW.

EDITOR'S NOTE

Some of you who have contributed articles may be wondering what is happening to them when they are not printed in the next Newsletter. Be patient!!! For one thing, there is a limited amount of pages that we can produce (because of cost) and also I have to have all the articles in at least 2 weeks prior to distribution. All the articles must be typed over by me so clear printing or double spaced typing would be appreciated.

If you have a particular program you wish produced, you can give me the program on tape and I will print it out for the newsletter. If it is a particularly long program, again because of the Newsletter limitations, it will probably be spread over 2 or more Newsletters.

If you are not particularly sure about certain areas of the ZX-81, either in programming or hardware, ask. Also if you want some areas explained fully in the Newsletter let me know.

MACHINE CODE PROGRAMMING

Before we can actually get involved with machine code, you must learn some of the commands that go with it. In the ZX-81 Basic language, we have roughly 45 commands; in the Z80 microprocessor, there are over 700!!! But, not only will we normally use less than a quarter of them, about three-quarters of the commands are duplications. So learning one command means you have automatically learned anywhere from 3 to 20 or 30 commands. So don't fret. It's really not that awful hard but you must think logically all the time and there are no Syntax checking capabilities.

In Basic, we have a powerful feature and that is we can assign anything to variables, numbers or characters: e.g. LET A = 10. That's

simple enough but let's see how much memory we use; first the line number takes 4 bytes, LET takes another byte, 2 more bytes for 'A' and '='. Because the ZX-81 is floating point arithmetic basic, the '1' in 10 takes 5 bytes and the '0' takes 1 byte. That makes a total of 13 bytes. Add to that the computer must go through the interpreter to check for Syntax and convert to machine code, then converts back again to Basic through the interpreter, it's no wonder that Basic is a lot slower (it seemed fast when you first got your computer) and is more wasteful of memory. The nice thing in Basic is that we can use any of the alpha characters or combinations, to assign the variables to. Alas, in machine code we are very restricted; we can use only 7 variables! They are A,B,C,D,E,H,L. (We also have the 'F' for the flags (to be explained) but we cannot use it for our variables.) With this restriction it seems next to impossible if we need to assign more than 7 variables. But, what we can do is Store that contents of that variable in some other part of memory and get it whenever we need it. That gives us a theoretical more than 16000 variables! We would never need to do this but you can see we have the capabilities.

Let's deal with the 'A' register (in machine code we call them registers instead of variables). The 'A' register is commonly called the 'ACCUMULATOR' since we can perform a lot of arithmetical manipulations with it whereas on the other register we don't have that capability; such as adding another number to one already existing in the 'A' register. We can't do that with the other registers. Although it seems we will be doing a lot of extra movements in comparison to Basic, machine code is very, very fast. For example, it can do roughly 10000 manipulations in less than a second! I think that will be fast enough for our needs.

We can, therefore, assign a number to the 'A' register the same as we did in the Basic variable 'A'.

```
LINE   HEX   DECIMAL   MNEMONIC

1      3E    62        LD A,n
```

In the above, the line number is used only to draw your attention to that line and is not ever used in machine code. (See the previous Newsletter about addresses are to machine code as line numbers are to Basic). Check in your manual and you will see the code for 62d or 3Eh is LD A,n. When you enter the decimal number 62, it means what it shows in the mnemonics (or opcodes) that we are assigning a number the variable register 'A' and the following number will be the assigned number. For example: 62, 10; means LD A, 10 or LOAD THE A REGISTER WITH NUMBER 10. (In basic, LET A=10). No machine code programmer can memorize all the machine code so you must keep checking the manual; although after some usage, will will memorize through habit, some of them.

The 'A' register now contains the numerical value of 10. If there were any other values assigned to 'A', they have disappeared. The same is true for the remaining registers. If you check the manual, you will see that the code for: LD B,n is 6. You can check the others yourself. If you read the last Newsletter, you will know that the highest number that can be assigned to one register is 255! Therefore, the Z80 chip have other codes that combine 2 registers together. They are in this form only: BC, DE, HL, (& AF). There are no other combinations.

These registers combined, behave in a different manner. To explain it easier we will use the HL register. With the registers combined, we

can assign numbers up to 65535. Each register (H & L) can still hold only numbers up to 255 but the computer recognizes it differently. Let's say we want to assign the number 10000 to the HL register. We must break it down so that neither register is holding a number larger than 255. (remember 0-255 are 256 numbers). Find the INT (10000/256) which is 39. Next multiply 39 * 256 which equals 9984. Subtract 10000-9984 which equals 16. We can then assign the 'high' byte (39) and the 'low' byte (16). Therefore, we would write the code in mnemonics as: LD HL, 10000. Checking the the machine code for this we find: 33 (decimal) is LD HL,nn. There is one more thing to complicated matters with combined registers. The computer must have the low byte first and the high byte second. Therefore, in actual fact, it is stored as 'L' first & 'H' second. We would therefore enter the bytes as: 33,16,39. The HL register now holds the value 10000. The same, again, holds true for all the other registers. You can see for the example, 'H' is high and 'L' is the low byte. Therefore, the BC register has the 'B' as the high byte and the 'C' as the low byte. DE is the same.

Enter a REM statement of 3 bytes and using your 'loader program' from the last Newsletter enter the address: 16514. Now enter 33 16 39. If you PEEK 16514, PEEK 16515, PEEK 16516, you will see they now contain your 3 bytes. (We will see in another Newsletter how to use the rgisters). Continuing on, check your manual and you will see we can add the contents of the registers together (e.g. code 129 is ADD A,C). You can see that we can also subtract registers (e.g. code 152 is SBC A,B) SBC is 'subtract with carry' (to be explained at another time). You read it as: SUBTRACT WITH CARRY, THE REGISTER 'B' FROM THE REGISTER 'A'. You will notice the codes from 144 to 151 has no other register after it. This is because that you can only subtract one register from the 'A' register and no others so 'A' is understood.

If you also check the other codes such as 66 decimal, it means LD B,D or load B with the value of the contents of the D register. One important thing to remember is that if you assigned a number such as 10000 to the HL register and later LD H with another value, you will have changed the contenst of the HL register.

I tried to make the above as clear as possible. You may find it clearer when we actually get to use them.

If you look back on what we've learned and check the manual for the various codes, you will see that by learning the above few machine codes, you will now know over 100 machine code commands!!!

There are a few more commands that are not that difficult to learn. If you look at the decimal code '12', you will see 'INC' which means to increase that register by a value of 1. If the value held by that register is 10 and the machine code routine comes across the instruction to 'INC', it will then increase that value to 11. Don't forget that the registers can hold only numbers up to 255 so if the register is already 255, then an INC command will make the value revert to a '0'. Note also the other command of 'DEC'. This is the same like INC except it means to decrease the value of that register by 1.

Well that's it for now. In the next newsletter I will show you how to use the above machine code commands in a small program.

SINGLE-KEY SHIFTED FUNCTIONS

CIRCUIT DESCRIPTION

Shifted functions require that the SHIFT key be held while striking the desired symbol/function key. It is evident from this that two circuits are involved, and that they must be activated simultaneously (or, at least in the specified sequence). It is also important that they be kept electrically isolated from each other to prevent "cross-talk" through the grid system.

The following paragraphs describe two methods for obtaining shifted functions with standard single-pole keyboard switches. The first is the Diode circuit, which is extremely simple, but which works for only the first seven functions listed in Table 1. However, these seven certainly include some of those that you will want.

The second method uses a Transmission Gate, which RCA calls a Quad Bilateral Switch; "Quad" because there are four in each package, and "Bilateral" because, when closed, they will pass signals in either direction.

THE DIODE METHOD

Figure 1 shows a Shifted-Function (S/F) key using the diode circuit. The figure also lists the seven functions for which it can be used. The desired function is determined by the selected data line on the keyboard to which Diode No. 2 is connected. This same circuit, using four "typical" keys, is shown in Figure 2.

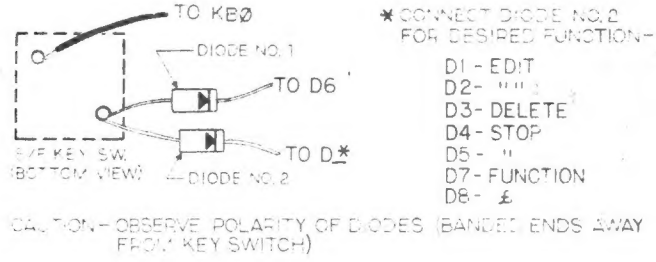

Figure 1. Single-Key Shifted Function Using Diodes

The diode method works only because the two circuits involved (the SHIFT circuit and each of the seven Function circuits) share the same KBØ output line (as shown in Table 1). The diodes are used to maintain isolation between the two "D" lines; diodes are one-way devices, and any signals that arrive at the key switch through one diode cannot go back out through the other. Because they are one-way, diodes are usually marked with a band at the "cathode" end. It is the end opposite the band which must be connected to the key switch terminal.

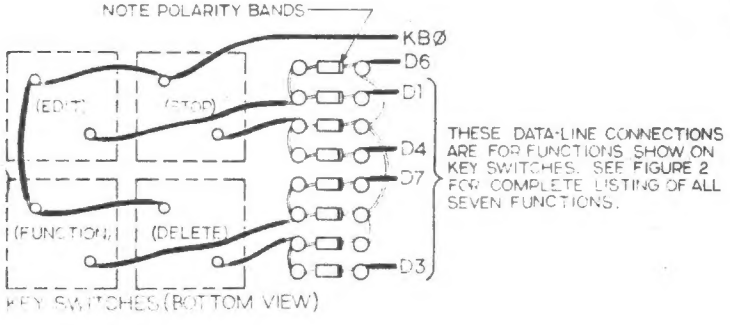

Figure 2. Typical Diode Circuit Wiring

BILATERAL SWITCH METHOD

This device is a non-mechanical, solid-state switch which closes when a control signal is applied. (See figure 3.) It is like a relay whose contacts close whenever power is applied to the magnetic coil. In this arrangement the keyboard switch no longer carries the data signals. It is used instead to supply the 5-volt control signal which holds the gates (switches) closed.

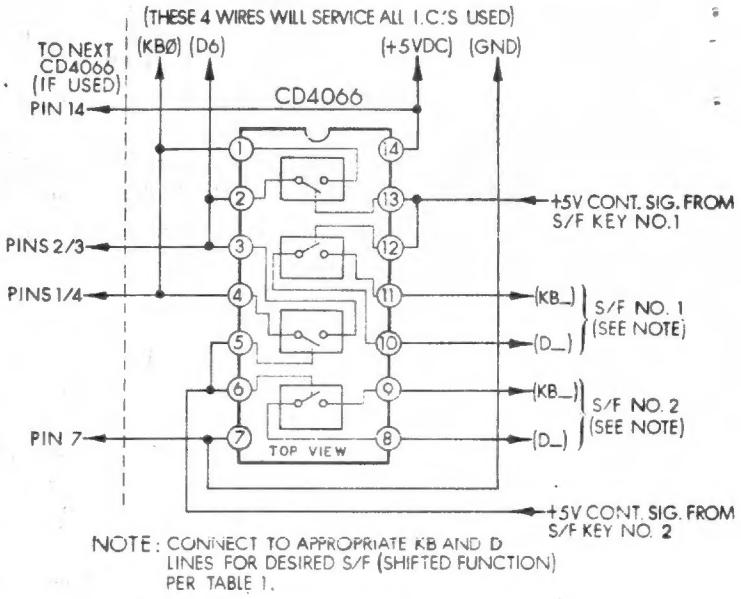

Figure 3. Two, Separate, Shifted-Function Circuits Using Quad Bilateral Switch CD4066

Each key uses two of the four bilateral switches in the IC package. Because each key uses one half of the IC, the other half may be used with another key for a second function. Then, too, several IC's can be added to the basic circuit. (Two are strung together in Figure 4.)

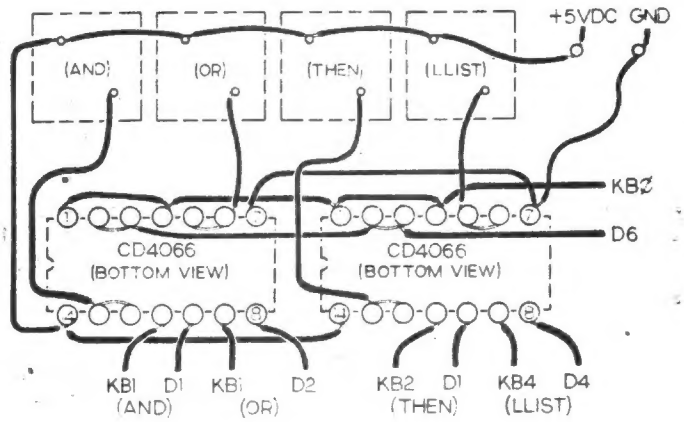

Figure 4. Typical Bilateral Switch Wiring

You will probably install the IC's on a separate mounting board, which can be attached to the keyboard with standoffs, screws and nuts, or with epoxy glue.

1st SWITCH CONNECTION	2nd SWITCH CONNECTION	FUNCTION OBTAINED		1st SWITCH CONNECTION	2nd SWITCH CONNECTION	FUNCTION OBTAINED
KBØ/D6 (SH)	KBØ/D1 (1)	EDIT		KBØ/D6 (SH)	KB2/D6 (X)	;
▲	KBØ/D2 (Q)	" "		▲	KB2/D7 (K)	+
	KBØ/D3 (Ø)	DELETE			KB2/D8 (M)	>
	KBØ/D4 (A)	STOP			KB3/D1 (4)	TO
	KBØ/D5 (P)	"			KB3/D2 (R)	<=
	KBØ/D7 (ENT)	FUNCTION			KB3/D3 (7)	↑
	KBØ/D8 (SP)	£			KB3/D4 (F)	FAST
	KB1/D1 (2)	AND			KB3/D5 (U)	$
	KB1/D2 (W)	OR			KB3/D6 (C)	?
	KB1/D3 (9)	GRAPHICS			KB3/D7 (J)	-
	KB1/D4 (S)	LPRINT			KB3/D8 (N)	<
	KB1/D5 (O)	)			KB4/D1 (5)	←
	KB1/D6 (Z)	:			KB4/D2 (T)	<>
	KB1/D7 (L)	=			KB4/D3 (6)	↓
	KB1/D8 (•)	;			KB4/D4 (G)	LLIST
	KB2/D1 (3)	THEN			KB4/D5 (Y)	>=
	KB2/D2 (E)	STEP			KB4/D6 (V)	/
	KB2/D3 (8)	→			KB4/D7 (H)	**
	KB2/D4 (D)	SLOW		KBØ/D6 (SH)	KB4/D8 (B)	*
KBØ/D6 (SH)	KB2/D5 (I)	(				

Table 1- Switch Connections for Shifted Functions

INSTALLATION

To operate the integrated circuits, power and ground wires must be added to the cable connecting the keyboard to the computer. If a 16-wire ribbon cable was used, two of the extra wires will handle the power requirements nicely. If two wires must be added to the cable, use the thinnest, flexible, insulated wires. They will be required to carry very little current.

The +5VDC and GND terminals on the computer board are indicated in Figure 5. At the keyboard end it is best to connect the power and ground wires to two insulated terminals. (If the keys are mounted on a PC board, the terminals can be a couple of brass screws in an unused area of the PC board.)

DIODE CIRCUITS. For any of the first seven functions listed in Table 1, you will certainly want to use the diode method, and save the IC's for the other 32 functions. If you wish, you can run the diodes directly to the data lines as shown in Figure 1, covering the entire "string" with insulated sleeving. However, especially if you have several keys that use the diode circuit, the layout shown in Figure 2 will be much neater. The diodes will take up very little space on any mounting board.

BILATERAL SWITCH CIRCUITS. The CD4066 IC's are CMOS devices, which are susceptible to damage from static electricity, before and during installation. This is the reason we recommend using DIP sockets, rather than soldering them directly into the circuit. Leave them in their conductive-foam packages until you are ready to install them. Avoid touching the terminals with your fingers - handle them by the ends of the plastic case. If the terminals must be bent inward to fit the socket, press the whole row against a flat metal surface.

The "typical" circuit shown in Figure 4 has several "unreal" qualities. First, the keys will not be located so conveniently, but remember, you can run the wires anywhere. Also, we show eight data lines at the bottom of the figure. Actually, you would want to conserve the number of wires to the keyboard, so the two KB1's would be connected together at the IC terminals, and a single line would serve both circuits. (The same applies to the two D1's, etc.)

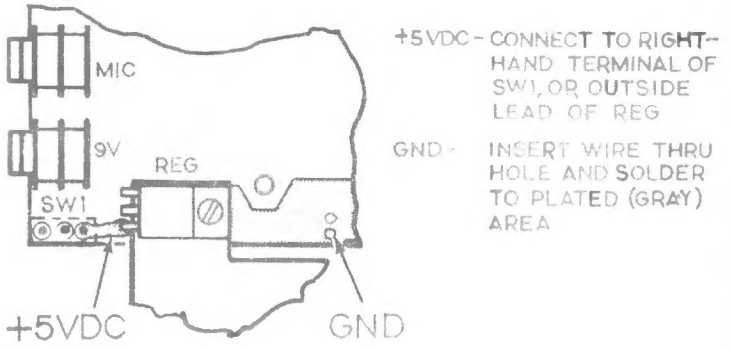

Figure 5. Power Connections at Computer Board

PARTS REQUIREMENTS

For reference, the following listing of the electronic components includes current Radio Shack catalog numbers and prices. You must determine the quantities.

Description	Type	Cat. No.	US $/Qty
Quad Bilateral Switch	CD4066	276-2466	.99
Diodes, Silicon	1N914	276-1122	.99/10
		276-1620	1.98/50
DIP IC Sockets	(14-pin)	276-1999	.89/2
Grid Board (0.1-inch hole-spacing) or equivalent	2-3/4" x 3-3/4"	276-158	1.95
	5-1/2" x 3-3/4"	276-161	2.95